OptiInfra Complete Windsurf Prompt Strategy v2.0

With Full Project Development Plan

Document Version: 2.0 COMPLETE
Date: October 16, 2025
Status: APPROVED - Ready for Execution
Total Prompts: 70 (5 Pilot + 65 Main)
Timeline: 11 weeks (8 weeks if parallel)
Current Phase: PILOT-01 ✅ COMPLETED

📋 TABLE OF CONTENTS

🎯 QUICK REFERENCE

Project Overview

OptiInfra = Multi-Agent AI Platform for LLM Infrastructure Optimization

- **Goal:** Cut costs 50%, improve performance 3x, ensure quality
- **Architecture:** 4 agents + 1 orchestrator + customer portal
- **Timeline:** 11 weeks (70 prompts)
- **Approach:** AI-assisted development with Windsurf

Current Status

✅ PILOT-01: Bootstrap Project Structure - COMPLETED
⏭️ NEXT: PILOT-02: Orchestrator Skeleton (Go)

Quick Stats

Metric	Value
Total Prompts	70
Phases	7 (Pilot + 6 main)
Duration	11 weeks sequential, 8 weeks parallel
Technologies	Go, Python, FastAPI, LangGraph, Next.js, PostgreSQL, ClickHouse, Qdrant, Redis
Target Savings	40-50% cost reduction
Target Performance	2-3x latency improvement

🌟 PROJECT GOALS & VISION

The Problem

Companies running LLM infrastructure (vLLM, TGI, SGLang) waste **$50K-$500K per month**:

- 40% waste from suboptimal configurations
- 50% GPU idle time ($30K-$40K/month per H100/A100)
- Over-provisioning due to fear of outages
- 20 hrs/week engineering time on manual optimization

The Solution

4 Intelligent Agents:

1. 💰 **Cost Agent**
   - Spot instance migration (30-40% savings)
   - Reserved instances (40-60% savings)
   - Right-sizing (20-30% savings)
   - Idle resource detection
2. ⚡ **Performance Agent**
   - KV cache tuning
   - Quantization (FP16→FP8→INT8)
   - Batch size optimization
   - Gradual rollout (10%→50%→100%)
3. 📊 **Resource Agent**
   - GPU utilization maximization
   - Auto-scaling (predictive)
   - Resource consolidation
   - KVOptkit integration
4. ✅ **Application Agent**
   - Quality monitoring (relevance, coherence, hallucinations)
   - Regression detection
   - A/B testing
   - Auto-rollback (if quality drops >5%)

Coordinated by:

- 🧠 **Orchestrator (Go)** - Agent registry, routing, conflict resolution, approval workflow

Value Proposition

- **Time to First Savings:** 2-3 weeks
- **Cost Reduction:** 40-50% proven
- **Performance Improvement:** 2-3x latency reduction
- **Quality Protection:** 95%+ regression detection
- **Team Time Saved:** 18 hrs/week (from 20 to 2)

📊 COMPLETE 70-PROMPT INVENTORY

Summary Table

Phase	Component	Prompts	Duration	Status
PILOT	Validation	5	3-5 days	P-01 ✅ / 4 ⬜
0	Foundation	15	1 week	⬜
1	Cost Agent	17	2 weeks	⬜
2	Performance Agent	11	2 weeks	⬜
3	Resource Agent	10	2 weeks	⬜
4	Application Agent	9	2 weeks	⬜
5	Portal & Production	8	1 week	⬜
	TOTAL	**70**	**11 weeks**	**1/70**

Detailed Prompt List

PILOT PHASE (Week 0) - 5 Prompts

#	Prompt Name	Purpose	Time	Dependencies	Status
P-01	Bootstrap Project	Create complete structure	30+20m	None	✅ DONE
P-02	Orchestrator Skeleton	Go HTTP server	30+20m	P-01	⬜ NEXT
P-03	Cost Agent Skeleton	FastAPI app	25+15m	P-01, P-02	⬜
P-04	LangGraph Setup	State machines	30+25m	P-03	⬜ CRITICAL
P-05	Spot Migration Workflow	End-to-end demo	40+30m	P-04	⬜

Decision Gate 1: After P-05 → ✅ GO / ⚠️ ADJUST / ❌ STOP

PHASE 0: FOUNDATION (Week 1) - 15 Prompts

Database Schemas (5 prompts):

#	Prompt	What It Creates	Time	Dependencies
0.2a	Core Schema	customers, agents, events, approvals	20+15m	P-01
0.2b	Cost Schema	cost_metrics, recommendations, optimizations	20+15m	0.2a
0.2c	Performance Schema	performance_metrics, slo_targets, violations	20+15m	0.2a
0.2d	Resource Schema	resource_metrics, scaling_events	20+15m	0.2a
0.2e	Application Schema	quality_metrics, baselines, regressions	20+15m	0.2a

Infrastructure (10 prompts):

#	Prompt	What It Creates	Time	Dependencies
0.3	ClickHouse Schema	Time-series tables, materialized views	15+10m	P-01
0.4	Qdrant Setup	Vector collections, embedding utils	15+10m	P-01
0.5	Orchestrator Skeleton	Already done in P-02	-	P-02
0.6	Agent Registry	Registration, health monitoring	15+10m	0.5
0.7	Request Routing	Load balancing, circuit breaker	20+15m	0.6
0.8	Coordination Logic	Conflict resolution, approval workflow	20+15m	0.7
0.9	Mock Cloud Provider	AWS/GCP/Azure simulator	15+10m	P-01
0.10	Shared Utilities	Database, config, logging utils	15+10m	0.2a
0.11	Monitoring	Prometheus + Grafana	25+20m	P-01, 0.10

Decision Gate 2: After 0.11 → ✅ Infrastructure ready?

PHASE 1: COST AGENT (Week 2-3) - 17 Prompts

#	Prompt	What It Creates	Time	Dependencies	Priority
1.1	Skeleton	Already done in P-03	-	P-03	-
1.2	AWS Collector	boto3, Cost Explorer API	25+20m	1.1, 0.2b	HIGH
1.3	GCP Collector	google-cloud, Billing API	20+15m	1.2	HIGH
1.4	Azure Collector	azure-sdk, Cost Management	20+15m	1.3	HIGH
1.5	LangGraph Setup	Already done in P-04	-	P-04	-
1.6	Spot Workflow	Already done in P-05	-	P-05	-
1.6b	Reserved Instance Workflow	RI recommendation workflow	30+25m	1.5, 1.2-1.4	HIGH
1.6c	Right-Sizing Workflow	Instance resizing workflow	30+25m	1.6b	HIGH
1.7	Analysis Engine	Idle detection, anomaly detection	30+25m	1.2-1.4	HIGH
1.8	LLM Integration	OpenAI/Anthropic client, prompts	25+20m	1.7	HIGH
1.9	Recommendation Engine	Prioritization, confidence scoring	25+20m	1.7, 1.8	HIGH
1.10	Execution Engine	Safe execution, rollback	30+25m	1.9	CRITICAL
1.11	Learning Loop	Qdrant storage, outcome tracking	25+20m	0.4, 1.10	MEDIUM
1.12	API Endpoints	All REST APIs	25+20m	1.1-1.11	HIGH
1.13	Unit Tests	80%+ coverage	35+25m	1.12	HIGH
1.14	Integration Tests	E2E workflow tests	30+25m	1.13	HIGH
1.14b	Performance Tests	Load testing, benchmarks	25+20m	1.14	MEDIUM
1.15	Documentation	Complete docs	25+20m	1.14b	MEDIUM

Decision Gate 3: After 1.15 → ✅ 40%+ savings demonstrated?

PHASE 2: PERFORMANCE AGENT (Week 4-5) - 11 Prompts

#	Prompt	What It Creates	Time	Dependencies
2.1	Skeleton	FastAPI app, registration	15+10m	0.5, 0.6, 0.10
2.2	vLLM Collector	Prometheus scraping, metrics	25+20m	2.1, 0.2c, 0.3
2.3	TGI Collector	TGI metrics collection	20+15m	2.2
2.4	SGLang Collector	SGLang metrics collection	20+15m	2.3
2.5	Analysis Engine	Bottleneck detection, SLO monitoring	30+25m	2.2-2.4
2.6	Optimization Engine	KV cache, quantization, batching	30+25m	2.5, 1.8
2.7	LangGraph Workflow	Gradual rollout workflow	30+25m	2.6, 1.5
2.8	API Endpoints	REST APIs	20+15m	2.1-2.7
2.9	Tests	Unit + Integration	35+25m	2.8
2.10	Performance Tests	Load testing	25+20m	2.9
2.11	Documentation	Complete docs	20+15m	2.10

PHASE 3: RESOURCE AGENT (Week 6-7) - 10 Prompts

#	Prompt	What It Creates	Time	Dependencies
3.1	Skeleton	FastAPI app, registration	15+10m	0.5, 0.6, 0.10
3.2	GPU Collector	nvidia-smi metrics	25+20m	3.1, 0.3
3.3	CPU/Memory Collector	psutil metrics	20+15m	3.2
3.4	Analysis Engine	Utilization analysis, scaling recs	30+25m	3.2, 3.3
3.5	KVOptkit Integration	KV cache optimization	25+20m	3.4
3.6	LangGraph Workflow	Resource optimization workflow	30+25m	3.5, 1.5
3.7	API & Tests	REST APIs + tests	30+25m	3.1-3.6
3.8	Performance Tests	Load testing	25+20m	3.7
3.9	Documentation	Complete docs	20+15m	3.8

PHASE 4: APPLICATION AGENT (Week 8-9) - 9 Prompts

#	Prompt	What It Creates	Time	Dependencies
4.1	Skeleton	FastAPI app, registration	15+10m	0.5, 0.6, 0.10
4.2	Quality Monitoring	Relevance, coherence, hallucination detection	30+25m	4.1, 0.2e, 0.3
4.3	Regression Detection	Baseline tracking, anomaly detection	30+25m	4.2
4.4	Validation Engine	A/B testing, approval/rejection	30+25m	4.3
4.5	LangGraph Workflow	Quality validation workflow	25+20m	4.4, 1.5
4.6	API & Tests	REST APIs + tests	30+25m	4.1-4.5
4.7	Performance Tests	Load testing	25+20m	4.6
4.8	Documentation	Complete docs	20+15m	4.7

PHASE 5: PORTAL & PRODUCTION (Week 10) - 8 Prompts

#	Prompt	What It Creates	Time	Dependencies
5.1	Next.js Setup	App Router, TailwindCSS, TypeScript	25+20m	0.1
5.2	Dashboard Components	All dashboards, charts, real-time	40+30m	5.1
5.3	Portal Tests	Playwright E2E tests	30+25m	5.2
5.4	Authentication	OAuth/JWT, RBAC	30+25m	5.1, ALL agents
5.5	Kubernetes Deployment	K8s manifests, Helm chart	35+30m	ALL services
5.6	CI/CD Pipeline	GitHub Actions	30+25m	5.5
5.7	API Security	Rate limiting, validation	25+20m	ALL agents
5.8	E2E System Tests	Complete system tests	40+30m	ALL prompts

Decision Gate 4: After 5.8 → ✅ Production ready?

🗺️ DEPENDENCY MAP & EXECUTION ORDER

Visual Dependency Flow

WEEK 0: PILOT (Must Complete Sequentially)

P-01 (Bootstrap) ✅ COMPLETED
↓
P-02 (Orchestrator Skeleton) ⏭️ NEXT
↓
P-03 (Cost Agent Skeleton)
↓
P-04 (LangGraph Setup) ⚠️ CRITICAL
↓
P-05 (Spot Workflow)
↓
DECISION GATE 1: ✅ Continue / ⚠️ Adjust / ❌ Stop

WEEK 1: FOUNDATION (Can Parallelize After 0.2a)

0.2a (Core Schema)
├──> 0.2b (Cost Schema)
├──> 0.2c (Performance Schema)
├──> 0.2d (Resource Schema)
├──> 0.2e (Application Schema)
└──> 0.10 (Shared Utilities)

P-01 (Bootstrap)
├──> 0.3 (ClickHouse)
├──> 0.4 (Qdrant)
├──> 0.9 (Mock Cloud)
└──> 0.11 (Monitoring)

P-02 (Orchestrator)
├──> 0.6 (Agent Registry)
│ ├──> 0.7 (Request Routing)
│ │ └──> 0.8 (Coordination)
│ │
│ └──> ALL Agent Skeletons (1.1, 2.1, 3.1, 4.1)
│
└──> DECISION GATE 2: ✅ Infrastructure Ready?

WEEK 2-3: COST AGENT (Sequential Within Agent)

1.1 (Skeleton) [from P-03]
↓
1.2 (AWS) → 1.3 (GCP) → 1.4 (Azure)
↓
1.7 (Analysis) → 1.8 (LLM) → 1.9 (Recommendations)
↓
1.10 (Execution) → 1.11 (Learning)
↓
1.12 (API) → 1.13 (Unit Tests) → 1.14 (Integration) → 1.14b (Perf Tests)
↓
1.15 (Docs)
↓
DECISION GATE 3: ✅ 40%+ Savings Proven?

Parallel to Cost Agent:
1.5 (LangGraph) [from P-04]
├──> 1.6 (Spot) [from P-05]
├──> 1.6b (Reserved Instance)
└──> 1.6c (Right-Sizing)

WEEK 4-9: OTHER AGENTS (Can Run in Parallel)

TEAM 1: Performance Agent (2.1 → 2.11)
TEAM 2: Resource Agent (3.1 → 3.9)
TEAM 3: Application Agent (4.1 → 4.8)

Each follows same pattern as Cost Agent
All depend on Foundation (0.x)
All use LangGraph (1.5)
All integrate with Orchestrator (0.6-0.8)

WEEK 10: PORTAL & PRODUCTION (Parallel After 5.1)

5.1 (Next.js Setup)

├──> 5.2 (Dashboards) → 5.3 (Portal Tests)
│
└──> 5.4 (Auth) [needs ALL agents]
↓
5.5 (Kubernetes) [needs ALL services]
↓
5.6 (CI/CD)
↓
5.7 (API Security) [needs ALL agents]
↓
5.8 (E2E Tests) [needs EVERYTHING]
↓
DECISION GATE 4: ✅ Production Ready?

Critical Path

The longest dependency chain (critical path):

P-01 → P-02 → P-03 → P-04 → P-05 → [Gate 1] →
0.2a → 0.10 → 1.1 → 1.2 → 1.3 → 1.4 → 1.7 → 1.8 → 1.9 →
1.10 → 1.11 → 1.12 → 1.13 → 1.14 → 1.14b → 1.15 → [Gate 3] →
2.1 → ... → 2.11 → 3.1 → ... → 3.9 → 4.1 → ... → 4.8 → [Gate 4] →
5.1 → 5.2 → 5.3 → 5.4 → 5.5 → 5.6 → 5.7 → 5.8

Total: 70 prompts
Duration: 11 weeks (sequential) or 8 weeks (parallel after Week 1)

✅ SUCCESS CRITERIA BY PHASE

PILOT Phase Success (Week 0)

After P-05, you must have:

Criteria	How to Verify	Target
All 5 prompts generate working code	Code compiles/runs	5/5 pass
Minimal manual fixes	% of code changed	<10%
LangGraph workflows execute	Run P-05 demo	Works end-to-end
Spot migration demo works	Calculate savings	Shows 30-40%
Can demonstrate to stakeholders	Live demo	10-min presentation

Validation Commands:


bash

```
# After P-02
cd services/orchestrator && go build && ./orchestrator
curl localhost:8080/health  # Should return 200

# After P-03
cd services/cost-agent && python src/main.py
curl localhost:8001/health  # Should return 200

# After P-04
cd services/cost-agent && pytest tests/test_langgraph.py

# After P-05
cd services/cost-agent && python -m src.workflows.spot_migration
# Should output: "Estimated savings: $18,000/month (38%)"
```

Decision: If 5/5 pass → ✅ GO to Foundation

Foundation Phase Success (Week 1)

After 0.11, you must have:

Criteria	How to Verify	Target
All databases operational	make verify	4/4 healthy
Orchestrator routes requests	Test API	Routes work
Monitoring dashboards visible	Open Grafana	Dashboards show data
All foundation tests pass	make test	100% pass
Database migrations run	Check Alembic	All applied

Validation Commands:


bash

```
make verify
# Expected:
# PostgreSQL... ✅ HEALTHY
# ClickHouse... ✅ HEALTHY
# Qdrant...     ✅ HEALTHY
# Redis...      ✅ HEALTHY

# Test orchestrator
curl localhost:8080/agents
# Should return list of registered agents

# Test monitoring
open http://localhost:3000  # Grafana
# Should see OptiInfra dashboards

# Test databases
psql postgresql://optiinfra:password@localhost:5432/optiinfra -c "SELECT * FROM customers;"
curl http://localhost:8123/ping  # ClickHouse
curl http://localhost:6333/collections  # Qdrant
```

Decision: If all pass → ✅ GO to Agents

Per-Agent Phase Success (Weeks 2-9)

After each agent (1.15, 2.11, 3.9, 4.8), you must have:

Criteria	How to Verify	Target
All workflows execute	Run E2E tests	Pass
80%+ test coverage	pytest --cov	≥80%
Can optimize real workloads	Manual test	Works
Measurable improvements	Run benchmarks	Targets met
Documentation complete	Review docs	All sections filled

Cost Agent Specific (1.15):

- ✅ Demonstrates 40-50% cost savings on test workload
- ✅ All 3 workflows work (spot, RI, right-sizing)
- ✅ LLM makes intelligent decisions
- ✅ Execution engine has rollback
- ✅ Learning loop stores outcomes

Performance Agent Specific (2.11):

- ✅ Demonstrates 2-3x latency improvement
- ✅ KV cache optimization works

- ✅ Gradual rollout mechanism (10%→50%→100%)
- ✅ Integrates with Application Agent (quality checks)

Resource Agent Specific (3.9):

- ✅ Demonstrates 2x GPU utilization improvement
- ✅ Auto-scaling works (predictive)
- ✅ KVOptkit integration functional

Application Agent Specific (4.8):

- ✅ Catches 95%+ quality regressions
- ✅ A/B testing works
- ✅ Auto-rollback triggers on quality drop >5%

Production Phase Success (Week 10)

After 5.8, you must have:

Criteria	How to Verify	Target
Portal accessible	Open browser	All pages load
Authentication works	Login test	OAuth succeeds
All dashboards show data	Check portal	Real-time updates
Kubernetes deployment works	kubectl get pods	All running
CI/CD pipeline functional	Git push	Auto-deploys
All E2E tests pass	Run 5.8 tests	100% pass

Validation Commands:

bash

```
# Portal
open http://localhost:3000
# Login with OAuth, view all dashboards

# Kubernetes
kubectl get pods -n optiinfra
# All pods should be Running

# CI/CD
git push origin main
# GitHub Actions should run and deploy

# E2E Tests
cd tests/e2e && pytest
# All 6 scenarios should pass
```

Decision: If all pass → ✅ DEPLOY to first customer

🚦 DECISION GATES

Gate 1: After Pilot (Week 0)

Question: Should we proceed with full 65-prompt development?

✅ PASS Criteria:

- 5/5 pilot prompts generated working code
- <10% manual code fixes needed
- LangGraph workflows execute correctly
- Can demonstrate spot migration with savings
- Team confident in Windsurf approach

Action: Generate remaining 65 prompts, proceed to Foundation

⚠️ ADJUST Criteria:

- 3-4 prompts work well
- 10-30% manual fixes needed
- Template needs refinement
- LangGraph works but needs syntax adjustment

Action:

1. Identify specific issues in prompts
2. Refine prompt template (add more context/examples)
3. Regenerate failing prompts
4. Re-test

5. Proceed once 5/5 pass

❌ **STOP Criteria:**

- <3 prompts generate working code
- 30% manual fixes required
- Windsurf doesn't understand LangGraph
- Fundamental architecture issues discovered

Action:

- **Option A:** Switch to manual development
- **Option B:** Simplify architecture (remove LangGraph)
- **Option C:** Try different AI tool (Claude Artifacts, Cursor)

Gate 2: After Foundation (Week 1)

Question: Is infrastructure ready for agent development?

✅ **PASS Criteria:**

- All 4 databases operational (PostgreSQL, ClickHouse, Qdrant, Redis)
- Orchestrator routes requests correctly
- Monitoring dashboards show metrics
- All foundation tests pass (100%)
- Database migrations run successfully

Action: Start agent development (Phase 1: Cost Agent)

⚠️ **ADJUST Criteria:**

- Most infrastructure works (3/4 databases)
- Some components need fixes
- Tests mostly pass (>80%)

Action:

1. Identify failing components
2. Debug issues (1-2 days)
3. Fix and re-test
4. Proceed once all pass

❌ **STOP Criteria:**

- Major infrastructure failures
- Databases don't start
- Orchestrator broken
- Cannot route requests

Action:

1. Deep debugging session
2. May need "Week 1.5" for fixes

3. Consider architecture simplification

Gate 3: After First Agent (Week 3)

Question: Is agent pattern proven? Can we scale to 3 more agents?

✅ **PASS Criteria:**

- Cost Agent fully functional
- Demonstrates 40%+ cost savings
- All tests pass (unit, integration, E2E)
- Code quality good (80%+ coverage)
- LLM integration works well
- Execution engine safe (has rollback)

Action: Proceed with remaining 3 agents (can parallelize)

⚠️ **ADJUST Criteria:**

- Agent mostly works
- Savings less than expected (20-30%)
- Some tests fail
- Code needs refactoring

Action:

1. Improve Cost Agent first
2. Update agent template based on learnings
3. Refine prompts for other agents
4. Proceed once Cost Agent solid

❌ **STOP Criteria:**

- Agent doesn't work
- No measurable value demonstrated
- Too many bugs
- Cannot demonstrate savings

Action:

1. Reassess agent architecture
2. May need to simplify LangGraph usage
3. Consider more manual development

Gate 4: Before Production (Week 9)

Question: Is system production-ready for customers?

✅ **PASS Criteria:**

- All 4 agents working correctly
- All E2E tests pass (100%)
- Measurable value demonstrated:
  - Cost: 40-50% savings

    - Performance: 2-3x improvement
    - Resource: 2x utilization
    - Application: 95%+ regression detection
- No critical bugs
- Documentation complete

Action: Deploy production essentials (Week 10)

⚠️ ADJUST Criteria:

- Most agents work (3/4)
- Some E2E tests fail
- Minor bugs present
- Value partially demonstrated

Action:

1. Fix failing agents
2. Debug E2E test failures
3. Address critical bugs
4. May need 1-2 extra days
5. Proceed once stable

❌ STOP Criteria:

- Major bugs in multiple agents
- Agents don't integrate well
- No measurable value
- E2E tests mostly fail

Action:

1. Extended testing/debugging phase
2. May need "Week 9.5" or "Week 10"
3. Reassess production timeline

📊 PROGRESS TRACKING CHECKLIST

How to Use This Checklist

1. Copy this section to a separate tracking document
2. Mark items as you complete them: ⬜ → 🔄 → ✅
3. Update weekly during team meetings
4. Use to report progress to stakeholders

Legend

- ⬜ Not started
- 🔄 In progress
- ✅ Completed
- ❌ Blocked
- ⚠️ Issues found

PILOT PHASE (Week 0)

Status: 1/5 completed (20%)

- [✅] P-01: Bootstrap Project Structure
- [⬜] P-02: Orchestrator Skeleton (Go)
- [⬜] P-03: Cost Agent Skeleton (FastAPI)
- [⬜] P-04: LangGraph Setup ← **CRITICAL**
- [⬜] P-05: Spot Migration Workflow

Decision Gate 1: [⬜] Evaluated - Result: ___

Notes:

- P-01 completed: All services running ✅
- Next: P-02 (Go orchestrator)

FOUNDATION PHASE (Week 1)

Status: 0/15 completed (0%)

Database Schemas:

- [⬜] 0.2a: Core Schema (customers, agents, events)
- [⬜] 0.2b: Cost Agent Schema
- [⬜] 0.2c: Performance Agent Schema
- [⬜] 0.2d: Resource Agent Schema
- [⬜] 0.2e: Application Agent Schema

Infrastructure:

- [⬜] 0.3: ClickHouse Time-Series Schema
- [⬜] 0.4: Qdrant Vector Database Setup
- [⬜] 0.6: Agent Registry
- [⬜] 0.7: Request Routing
- [⬜] 0.8: Coordination Logic
- [⬜] 0.9: Mock Cloud Provider
- [⬜] 0.10: Shared Python Utilities
- [⬜] 0.11: Monitoring (Prometheus + Grafana)

Decision Gate 2: [⬜] Evaluated - Result: ___

COST AGENT (Week 2-3)

Status: 0/17 completed (0%)

Collectors:

- [⬜] 1.2: AWS Cost Metrics Collector
- [⬜] 1.3: GCP Cost Metrics Collector

- [ ] 1.4: Azure Cost Metrics Collector

Workflows:

- [ ] 1.6b: Reserved Instance Workflow
- [ ] 1.6c: Right-Sizing Workflow

Core Logic:

- [ ] 1.7: Analysis Engine
- [ ] 1.8: LLM Integration
- [ ] 1.9: Recommendation Engine
- [ ] 1.10: Execution Engine ← **CRITICAL**
- [ ] 1.11: Learning Loop

API & Testing:

- [ ] 1.12: API Endpoints
- [ ] 1.13: Unit Tests (Target: 80%+ coverage)
- [ ] 1.14: Integration Tests
- [ ] 1.14b: Performance Tests
- [ ] 1.15: Documentation

Metrics:

- Cost Savings Demonstrated: ___%
- Test Coverage: ___%
- Workflows Passing: __/3

Decision Gate 3: [ ] Evaluated - Result: ___

PERFORMANCE AGENT (Week 4-5)

Status: 0/11 completed (0%)

- [ ] 2.1: Skeleton
- [ ] 2.2: vLLM Metrics Collector
- [ ] 2.3: TGI Metrics Collector
- [ ] 2.4: SGLang Metrics Collector
- [ ] 2.5: Analysis Engine
- [ ] 2.6: Optimization Engine
- [ ] 2.7: LangGraph Workflow
- [ ] 2.8: API Endpoints
- [ ] 2.9: Tests
- [ ] 2.10: Performance Tests
- [ ] 2.11: Documentation

Metrics:

- Latency Improvement: __x
- Test Coverage: ___%

RESOURCE AGENT (Week 6-7)

Status: 0/10 completed (0%)

- [ ] 3.1: Skeleton
- [ ] 3.2: GPU Utilization Collector
- [ ] 3.3: CPU/Memory Collector
- [ ] 3.4: Utilization Analysis Engine
- [ ] 3.5: KVOptkit Integration
- [ ] 3.6: LangGraph Workflow
- [ ] 3.7: API & Tests
- [ ] 3.8: Performance Tests
- [ ] 3.9: Documentation

Metrics:

- GPU Utilization Improvement: __x
- Test Coverage: ___%

APPLICATION AGENT (Week 8-9)

Status: 0/9 completed (0%)

- [ ] 4.1: Skeleton
- [ ] 4.2: Quality Monitoring
- [ ] 4.3: Regression Detection
- [ ] 4.4: Validation Engine
- [ ] 4.5: LangGraph Workflow
- [ ] 4.6: API & Tests
- [ ] 4.7: Performance Tests
- [ ] 4.8: Documentation

Metrics:

- Regression Detection Rate: ___%
- Test Coverage: ___%

Decision Gate 4: [ ] Evaluated - Result: ___

PORTAL & PRODUCTION (Week 10)

Status: 0/8 completed (0%)

- [ ] 5.1: Next.js Portal Setup
- [ ] 5.2: Dashboard Components
- [ ] 5.3: Portal Integration Tests
- [ ] 5.4: Authentication (OAuth/JWT)
- [ ] 5.5: Kubernetes Deployment

- [ ] 5.6: CI/CD Pipeline
- [ ] 5.7: API Security & Rate Limiting
- [ ] 5.8: E2E System Tests

Production Readiness:

- [ ] All services deployed to K8s
- [ ] Authentication working
- [ ] CI/CD pipeline functional
- [ ] All E2E tests passing
- [ ] Ready for first customer

OVERALL PROGRESS

Completion Status:

Total Prompts: 70
Completed: 1
In Progress: 0
Remaining: 69
Percentage: 1.4%

Phase Completion:

✅ PILOT: 1/5 (20%) 🔄 IN PROGRESS
⬜ Foundation: 0/15 (0%)
⬜ Cost Agent: 0/17 (0%)
⬜ Performance: 0/11 (0%)
⬜ Resource: 0/10 (0%)
⬜ Application: 0/9 (0%)
⬜ Production: 0/8 (0%)

Decision Gates:

- [ ] Gate 1: Pilot evaluation (Week 0)
- [ ] Gate 2: Foundation ready (Week 1)
- [ ] Gate 3: First agent proven (Week 3)
- [ ] Gate 4: Production ready (Week 9)

TIMELINE & MILESTONES

Gantt Chart View

Week 0 (PILOT): [████░░░░░░░░░░░░░░░░] 20% (P-01 done)
Week 1 (Foundation): [░░░░░░░░░░░░░░░░░░░░] 0%
Week 2-3 (Cost): [░░░░░░░░░░░░░░░░░░░░] 0%
Week 4-5 (Perf): [░░░░░░░░░░░░░░░░░░░░] 0%
Week 6-7 (Resource): [░░░░░░░░░░░░░░░░░░░░] 0%
Week 8-9 (App): [░░░░░░░░░░░░░░░░░░░░] 0%
Week 10 (Production): [░░░░░░░░░░░░░░░░░░░░] 0%

Detailed Weekly Plan

Week 0: PILOT (Current Week)

Goal: Validate AI-assisted development approach

Daily Schedule:

- **Day 1 (Completed):** ✅ P-01 Bootstrap
- **Day 2 (Next):** P-02 Orchestrator + P-03 Cost Agent Skeleton
- **Day 3:** P-04 LangGraph Setup + P-05 Spot Workflow (morning)
- **Day 3 (afternoon):** End-to-end testing
- **Day 4:** Bug fixes, refinements
- **Day 5:** Decision Gate 1 evaluation, prepare for Week 1

Deliverable: Working demo showing 30-40% cost savings potential

Key Milestone: 🎯 Prove Windsurf can generate production code

Week 1: FOUNDATION

Goal: Build infrastructure for all agents

Daily Schedule:

Monday (Day 1):

- Morning: 0.2a Core Schema (20+15m)
- Morning: 0.2b Cost Schema (20+15m)
- Afternoon: 0.2c Performance Schema (20+15m)
- Afternoon: 0.2d Resource Schema (20+15m)
- End of day: 0.2e Application Schema (20+15m)
- Verify: All migrations run

Tuesday (Day 2):

- Morning: 0.3 ClickHouse Schema (15+10m)
- Morning: 0.4 Qdrant Setup (15+10m)
- Afternoon: 0.10 Shared Utilities (15+10m)
- Afternoon: 0.9 Mock Cloud Provider (15+10m)
- Verify: All databases queryable

Wednesday (Day 3):

- Morning: 0.6 Agent Registry (15+10m)
- Morning: 0.7 Request Routing (20+15m)
- Afternoon: 0.8 Coordination Logic (20+15m)
- Verify: Orchestrator fully functional

Thursday (Day 4):

- Morning: 0.11 Monitoring Setup (25+20m)
- Afternoon: Integration testing
- Verify: Prometheus + Grafana showing metrics

Friday (Day 5):

- Morning: Run all foundation tests
- Afternoon: Decision Gate 2 evaluation
- Prepare for Week 2 (Cost Agent)

Deliverable: Complete infrastructure ready for agents

Key Milestone: 🎯 All 4 databases + orchestrator + monitoring operational

---

Week 2-3: COST AGENT

Goal: Complete cost optimization agent

Week 2 Schedule:

Monday: 1.2 AWS + 1.3 GCP + 1.4 Azure Collectors **Tuesday:** 1.6b Reserved Instance + 1.6c Right-Sizing Workflows **Wednesday:** 1.7 Analysis Engine + 1.8 LLM Integration **Thursday:** 1.9 Recommendation Engine + 1.10 Execution Engine **Friday:** 1.11 Learning Loop + 1.12 API Endpoints

Week 3 Schedule:

Monday: 1.13 Unit Tests (ensure 80%+ coverage) **Tuesday:** 1.14 Integration Tests + 1.14b Performance Tests **Wednesday:** 1.15 Documentation **Thursday:** E2E testing, bug fixes **Friday:** Decision Gate 3 evaluation, demonstrate savings

Deliverable: Cost Agent demonstrating 40-50% savings

Key Milestone: 🎯 First agent fully functional with measurable value

---

Week 4-5: PERFORMANCE AGENT

Goal: Performance optimization agent

Schedule: Similar to Cost Agent (2 weeks)

Week 4:

- Mon-Tue: Collectors (2.2-2.4)
- Wed-Thu: Analysis + Optimization (2.5-2.6)
- Fri: Workflow (2.7)

Week 5:

- Mon: API (2.8)
- Tue: Tests (2.9-2.10)
- Wed: Documentation (2.11)
- Thu-Fri: Testing, validation

Deliverable: Performance Agent showing 2-3x improvement

Key Milestone: 🎯 Second agent working, pattern validated

Week 6-7: RESOURCE AGENT

Goal: Resource optimization agent

Schedule: Similar pattern (2 weeks, but 10 prompts vs 11)

Deliverable: Resource Agent showing 2x utilization

Key Milestone: 🎯 Third agent complete

Week 8-9: APPLICATION AGENT

Goal: Quality monitoring agent

Schedule: Similar pattern (2 weeks, 9 prompts)

Deliverable: Application Agent catching 95%+ regressions

Key Milestone: 🎯 All 4 agents complete, Decision Gate 4

Week 10: PORTAL & PRODUCTION

Goal: Production-ready system

Daily Schedule:

Monday: 5.1 Portal Setup + 5.2 Dashboard Components **Tuesday:** 5.3 Portal Tests + 5.4 Authentication **Wednesday:** 5.5 Kubernetes Deployment + 5.6 CI/CD **Thursday:** 5.7 API Security + 5.8 E2E System Tests **Friday:** Final validation, prepare for customer deployment

Deliverable: Production-ready OptiInfra

Key Milestone: 🎯 Ready to onboard first design partner

Critical Milestones

Week	Milestone	Success Metric
0	Pilot Complete	Demo works, 30-40% savings shown
1	Infrastructure Ready	All services healthy
3	First Agent Proven	40%+ savings demonstrated
5	Pattern Validated	2nd agent works, faster than 1st
7	3 Agents Complete	All optimizations working
9	4 Agents Complete	Quality protected
10	Production Ready	Can deploy to customer
11+	First Customer	Design partner onboarded

🛡️ RISK MANAGEMENT

High Priority Risks

Risk 1: Windsurf Generates Poor Quality Code

Likelihood: Medium
Impact: High
Phase: All phases

Mitigation:

- ✅ Pilot phase validates Windsurf first (Week 0)
- ✅ Detailed prompts with examples and context
- ✅ Clear success criteria for each prompt
- ✅ Validation commands after each prompt
- ✅ Human review before integration

Contingency:

- If pilot fails: Switch to manual development or different AI tool
- If quality issues persist: Increase human review, simplify prompts
- Budget 10% time for manual code refinement

Status: ⏳ Testing in Pilot phase

Risk 2: LangGraph Too Complex for Windsurf

Likelihood: Medium
Impact: High
Phase: PILOT-04, all agent workflows

Mitigation:

- ✅ Test LangGraph specifically in P-04 (early detection)
- ✅ Provide complete LangGraph examples in prompts
- ✅ Include state diagrams and pseudocode

- ✅ Have fallback: plain Python state machines

Contingency:

- If P-04 fails: Simplify to plain Python (less elegant, but works)
- Alternative: Use existing workflow libraries (Temporal, Prefect)
- Budget 1 week for architecture pivot if needed

Status: ⏳ Will test in P-04 (Day 3)

Risk 3: Database Schema Becomes Unwieldy

Likelihood: Low
Impact: Medium
Phase: Foundation (0.2a-0.2e)

Mitigation:

- ✅ Already split into 5 separate prompts (manageable)
- ✅ Clear relationships defined between schemas
- ✅ Incremental migrations (one prompt at a time)

Contingency:

- If schema conflicts arise: Create additional migration prompts
- Can always refactor schemas later (Alembic supports this)

Status: ✅ Mitigated by design

Risk 4: Integration Issues Between Components

Likelihood: Medium
Impact: Medium
Phase: All phases, especially Week 10

Mitigation:

- ✅ Clear API contracts defined in prompts
- ✅ Extensive integration testing (prompts 1.14, 2.9, etc.)
- ✅ E2E system test (prompt 5.8)
- ✅ Mock services for isolated testing

Contingency:

- If integration fails: Create additional integration prompts
- Use API versioning for backward compatibility
- Budget 1-2 days per phase for integration debugging

Status: ⏳ Will monitor throughout

Risk 5: Timeline Slippage

Likelihood: High
Impact: Medium
Phase: All phases

Mitigation:

- ✅ Buffer time built into each prompt (20-30 min verification)
- ✅ Parallel execution possible after Week 1
- ✅ MVP approach: Can ship with 1-2 agents initially
- ✅ Weekly progress reviews (adjust as needed)

Contingency:

- If behind schedule: Parallelize agent development (4 teams)
- Defer non-critical features (monitoring, auth can be basic)
- Focus on demonstrating value (savings) first

Status: ⏳ Week 0 on track (1/5 prompts done Day 1)

Risk 6: Production Deployment Issues

Likelihood: Medium
Impact: High
Phase: Week 10

Mitigation:

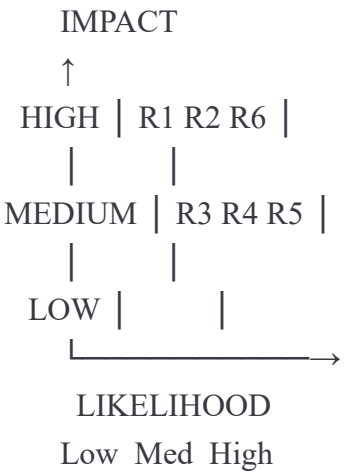
- ✅ Comprehensive testing throughout (unit, integration, E2E, performance)
- ✅ Staging environment (test before production)
- ✅ Gradual rollout mechanism built-in
- ✅ Rollback capability in execution engine
- ✅ Monitoring from Day 1 (Prometheus/Grafana)

Contingency:

- If production issues: Rapid rollback (< 5 minutes)
- Use design partner program (friendly first customers)
- Have 24/7 on-call during initial deployment

Status: ⏳ Will prepare in Week 10

Risk Matrix



R1 = Risk 1 (Code Quality)
R2 = Risk 2 (LangGraph)
R3 = Risk 3 (Database)
R4 = Risk 4 (Integration)
R5 = Risk 5 (Timeline)
R6 = Risk 6 (Production)

📖 HOW TO USE THIS DOCUMENT

Daily Development Workflow

1. Start of Day:

□ Check "Timeline & Milestones" section
□ Identify today's prompts (e.g., "Monday Week 1: 0.2a, 0.2b, 0.2c")
□ Review dependencies for each prompt
□ Ensure prerequisites are met

2. For Each Prompt:

□ Find prompt in "Complete 70-Prompt Inventory"
□ Come to Claude chat and request: "Generate detailed prompt for X.Y"
□ Copy detailed prompt from Claude
□ Paste into Windsurf
□ Wait for code generation
□ Run validation commands
□ Check success criteria
□ Mark as complete in "Progress Tracking Checklist"

3. End of Day:

□ Update "Progress Tracking Checklist"
□ Update "Timeline & Milestones" (Gantt chart)
□ Note any blockers or issues
□ Prepare for tomorrow's prompts

Weekly Planning

Every Friday:

□ Review week's progress
□ Calculate completion percentage
□ Check if on track for milestones
□ Evaluate any risks
□ Plan next week's prompts
□ Hold team meeting (if parallel development)

Decision Gate Process

At Each Gate (Week 0, 1, 3, 9):

□ Run all validation commands
□ Review all success criteria
□ Calculate metrics (savings, performance, coverage)
□ Make GO / ADJUST / STOP decision
□ Document decision and rationale
□ If GO: Proceed to next phase
□ If ADJUST: Create action plan, timeline
□ If STOP: Evaluate alternatives

When You Need Help

If a Prompt Fails:

1. Check "Troubleshooting" section in the prompt
2. Run validation commands again
3. Check dependencies (did prereq prompts succeed?)
4. Come back to Claude and say:
   "Prompt X.Y failed with error: [paste error]
   Validation output: [paste output]
   What should I fix?"
5. I'll help debug or regenerate the prompt

If Behind Schedule:

1. Check "Risk Management" section
2. Review "Timeline & Milestones"
3. Identify bottleneck (which prompts taking longest?)
4. Consider parallel execution (if Week 1 complete)
5. Defer non-critical features if needed
6. Update timeline, communicate to team

If Need to Skip Ahead:

1. Check "Dependency Map" - ensure prerequisites done
2. Some prompts CAN be done in parallel:
   - After 0.2a: All other database schemas (0.2b-0.2e)
   - After Week 1: All 4 agents (if 4 teams)
   - After 5.1: Portal components (5.2, 5.3, 5.4)
3. Never skip: PILOT prompts, Decision Gates, Critical Path

Document Maintenance

Update This Document When:

- □ Prompt fails and needs to be split (update inventory)
- □ New prompt needed (add to inventory and dependency map)
- □ Timeline changes (update Gantt chart)
- □ Risk occurs (update risk status)
- □ Milestone achieved (mark in timeline)
- □ Technology changes (update tech stack references)

🎯 QUICK COMMAND REFERENCE

Most Used Commands

bash

```
# Daily workflow
make verify          # Check all services healthy
make logs            # View logs for debugging
make test            # Run all tests
git add . && git commit -m "Prompt X.Y complete"

# Start of day
make up              # Start all services
docker ps            # Check what's running

# End of day
make down            # Stop all services (optional)

# Validation
pytest --cov         # Check test coverage
pytest tests/ -v     # Run tests with verbose output
curl localhost:8080/health  # Check orchestrator
curl localhost:8001/health  # Check cost agent

# Database
psql $DATABASE_URL -c "SELECT * FROM customers;"
curl http://localhost:8123/ping  # ClickHouse
curl http://localhost:6333/collections  # Qdrant

# Monitoring
open http://localhost:3000  # Grafana dashboards
open http://localhost:9090  # Prometheus

# Kubernetes (Week 10)
kubectl get pods -n optiinfra
kubectl logs <pod-name> -n optiinfra
kubectl describe pod <pod-name> -n optiinfra
```

📞 GETTING HELP

From Claude (Me!)

Ask me for:

- Detailed prompts: "Generate prompt for X.Y"
- Debugging help: "Prompt X.Y failed with error: ..."

- Clarifications: "What does prompt X.Y create exactly?"
- Strategy questions: "Should I parallelize Week 4-9?"

From Documentation

Check these sections:

- **Prompt Inventory:** What each prompt does
- **Dependency Map:** What order to execute
- **Success Criteria:** How to validate
- **Risk Management:** Common issues and solutions
- **Timeline:** When things should happen

🎉 NEXT STEPS

Right Now (You Are Here)

Current Status:

- ✅ PILOT-01 completed
- ⏭️ Ready for PILOT-02

Your Next Actions:

1. **Save this document** as reference throughout project
2. **Come back to Claude** and say: "Generate PILOT-02"
3. **Execute PILOT-02** in Windsurf
4. **Validate** using success criteria
5. **Continue** to PILOT-03, 04, 05
6. **Hit Decision Gate 1** after PILOT-05

📊 DOCUMENT SUMMARY

This document contains:

- ✅ Complete 70-prompt inventory with details
- ✅ Dependency map (execution order)
- ✅ Success criteria for each phase
- ✅ 4 decision gates with criteria
- ✅ Progress tracking checklist
- ✅ 11-week timeline with daily schedules
- ✅ Risk management strategies
- ✅ How to use guide
- ✅ Quick command reference

Use this document to:

- ✅ Understand the entire project
- ✅ Track progress (update checklist weekly)

- ✅ Know what's next (check timeline)
- ✅ Make decisions (use decision gates)
- ✅ Get unstuck (use risk management)
- ✅ Stay on track (follow timeline)

Last Updated: October 16, 2025
Next Review: After PILOT-05 (Decision Gate 1)
Status: IN PROGRESS - Week 0, Day 1
Completion: 1/70 prompts (1.4%)

🚀 **Let's build OptiInfra!**